Towards Programmable and Scalable IoT Infrastructures for Smart Cities

Ancuta Corici, Ronald Steinke, Thomas Magedanz
Next Generation Network Infrastructures
Fraunhofer FOKUS
Berlin, Germany
{andreea.ancuta.corici, ronald.steinke, tm}@fokus.fraunhofer.de

Louis Coetzee, Dawid Oosthuizen, Buhle Mkhize
CSIR Meraka Institute, CSIR
Pretoria, South Africa
{louis.coetzee, doosthuizen, bmkhize}@csir.co.za

Marisa Catalan, Jacint Castells Fontelles, Josep Paradells
I2CAT
Barcelona, Spain
{marisa.catalan, josep.paradells}@i2cat.net, jacint@ebre.cat

Ranjan Shrestha, Daniel Nehls, Bjoern Riemer
AV Department
Technical University
Berlin, Germany
{ranjan.shrestha, daniel.nehls, bjoern.riemer}@tu-berlin.de

***Abstract*—Smart Cities applications and infrastructures are actively being developed and rolled out. However, maintenance complexity is significant, often limiting deployments to small regions or small cities. To support gradual or spontaneous infrastructure scaling at region or national levels, infrastructure management that monitors end device connectivity and ensures overall IoT communication reliability becomes key. This article describes a method using intercontinental research facilities that programmatically manages smart devices and their communication with the ultimate aim to elastically deploy IoT servers in the cloud. Implementation details and experimental results of real devices are included.**

Keywords— IoT; Smart Cities; OMA LWM2M; NFV; FIRE; CoAP

I. INTRODUCTION

Deploying Smart City solutions in scaled environments implies the transfer of laboratory-ready technologies to the real world. Moving from a contained "Smart City" environment to a real-world deployment is challenging. Amongst others, aspects related to security, identity, addressability and latency become more complex and often require manual intervention as the number of devices and services increase. To bridge the gap between a small environment and one for a large city, an experimental facility with associated services is an important mechanism to understand the challenges. Afterwards, large-scale infrastructures can be rolled out with confidence.

Context and country specific research facilities enable scientists to validate their work. Experimental research facilities for the developed world already exist (e.g. FIRE GENI [21]). They provide a means to experiment with architectures that scale beyond the laboratory [22].

Combined facilities (for instance testbeds in a developed world federated with similar ones from a developing world) support idea cross-pollination and learning. Such environments cover a variety of challenge types that, when resolved, will also help overcome scaled deployment challenges.

In contrast to the developed world, experimental research facilities in developing countries are not common. To address this, intercontinental research facilities bootstrap developing countries to accelerate finding answers to questions. They also provide insights into problems beyond the traditional and well-understood context. Testbeds of this nature allow for cross-domain services data and services integration (i.e. smart environmental sensing and smart energy in different countries). The learnings from geospatially distributed federated facilities contribute to easing deployment of scaled developments (i.e. in a city or a small region).

With the increase in the number of devices, the velocity, variety, and volume of data also increase. This implies that the experimental facility's back-end needs to scale accordingly. Related work [1] proposes methods using Platform as a Service (PaaS). To cater for back-end service elastic deployment, maintenance or a potential cyberattack that would render one of the services unresponsive, the smart devices deployed in the field have to adapt rapidly to the infrastructure and use newly assigned service instances.

TRESCIMO [7] is an example of an intercontinental research project that has built a facility to enable M2M experiments and limited area trials for smart energy and environments that ran during the project. The facility is well positioned for PaaS IoT experiments to address IoT deployments challenges resembling those faced in scaled, real-world environments. This paper presents requirements analysis for large-scale deployments in section II, background in section III and the concept developed in the TRESCIMO project in section IV, followed by the architecture and implementation extensions in sections V and VI respectively. Through experiments and the analysed results presented in section VII, the paper also investigates the suitability of PaaS when applied to a large, distributed context with heterogeneous data sources. Aspects such as continued data flow after back-end service redeployment, the impact of geographic services distribution, and the impact of network latency are investigated. Section VIII concludes and highlights future work.

II. Requirements and Constraints Analysis

Reliable large-scale IoT systems serving different domains over a large geographic distribution have the following key technical requirements: **configurability, scalability, security and energy efficiency** of the infrastructure. **Interoperability** with other systems has to be considered by using open standards, thereby making it more likely to integrate with other market components.

Infrastructure management cost efficiency is an important requirement for allowing a prototype to evolve from a trial complexity of several/tens of entry points (buses, bus stops, households) to the complexity of a metropole or an entire country the size of Spain or South Africa. To address this requirement, methods such as service and device discovery and updating are required.

Appropriate research facilities are needed for experimentation and validation of concepts related to large-scale Smart City deployments. The TRESCIMO Future Internet Research and Experimentation (FIRE) testbed which is federated over continents is such an environment. Using published FIRE APIs, scientists are able to experiment and use measurements for decision timing with gradual or abrupt infrastructure scaling operations.

A Smart City architecture should support at least two domains to properly cater for the diversity found in Smart City data. This requirement validates whether the solution is sufficiently general for it to be reused in other domains and contexts not already included in the current solution.

III. Background

A. Related Work

The idea of IoT PaaS for scaling is evolving rapidly with solutions based on load-balancing already taking shape [1]. Although load balancing and distribution can leverage the new virtualization and software enabled services, it can lead to loss of context as data associated with the same vertical might end up too distributed for real-time analysis and utilization.

The IoT6 project proposed a solution for scaling and adaptive IoT infrastructures that incorporates multicast DNS, DNS – Service-Discovery (DNS-SD) and CoAP [5] for registration and access of resources and services from IoT devices [2]. The same approach can provide end-to-end scalability; that is, from the device to the back-end servers running in the cloud. This mechanism leaves the decision of which service instance to send the data to, to the device. Although this is feasible for static back-end servers, in the case of dynamically deployed back-end servers the smart device sending M2M data would need to perform a DNS service discovery every time it sends or retrieves data in order to update the DNS cache. Our approach ensures that the device is notified about which back-end server to use and therefore the device does not continuously need to look up the service instance properties.

The concept of programmatically enabled networks was recently introduced as an umbrella over Software Defined Networks and Flexible/on-demand Service Deployments [3]. We believe that the true potential of this concept can only be achieved if smart devices are integrated as programmable devices as far as their communication is concerned.

B. Protocols and Standards

The Constrained Application Protocol (CoAP) [5] was designed for low power and constrained networks. CoAP uses UDP to lower communication overheads with an optional delivery confirmation system. It follows a RESTful architectural style. GET, PUT, POST and DELETE methods are used with a URI to access a resource. Observation of a specific resource is implemented using a publish/subscribe approach. CoAP relies on the CoRE link format (RFC6690) [6] for resource discovery and description. Through resource discovery, the resource URIs are published via the ".well-known/core/" URI and meta-information describe attributes of a resource, i.e. the resource and content types. A resource directory can also be created and published using this format.

Lightweight M2M (LW M2M) version 1.0 [4] is a new technical standard for remote management of M2M devices and is based on CoAP [5] and CoRE link formats [6]. The architecture comprises a device management (DM) client that resides on the M2M device as well as a DM server that is located on the M2M Service Provider or the Network Provider as shown in Fig. 1. The LWM2M protocol is suitable for complex and powerful computing devices and can handle constrained devices and works well in unreliable networks. The predefined set of objects includes Access Control, Connectivity Monitoring and Statistics, Device, Firmware Update and Location management. In order to extend the model, organizations can define and propose new objects that could be important for a robust device management platform. During registration at the server, the LW M2M client communicates a unique endpoint client name as well as the list of resources hosted at that client.

The ETSI NFV MANO [20] standard specifies how Network Functions can be deployed on demand. The set of Network Functions include Mobile Core Network, IP Multimedia Subsystem and recently IoT components. There is a deep synergy with concepts of Software Defined Networking (SDN) in which protocols for managing the data path of the network has been defined. The impact that SDN and NFV technologies have on IoT infrastructures can be strengthened by interfacing with M2M technologies such as CoAP and LWM2M to ease the adoption.

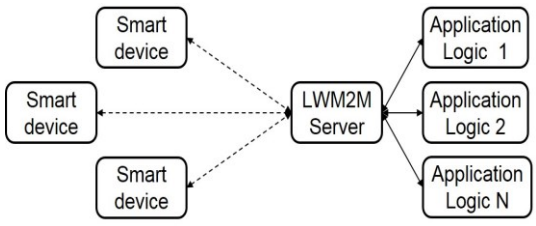

Fig. 1: LWM2M protocol

IV. Concept

In order for an IoT infrastructure to become end-to-end programmable, support for flexible runtime communication capabilities and adaptation is required on multiple levels. This concept is depicted in Fig. 2 and the three layers are described next. The Programmable Smart Device layer is comprised of smart devices that can be activated, deactivated, or accept configuration updates about the endpoint. At the Flexible Logic layer, smart devices can be matched to currently available services using a Smart Device Communication Manager. The final layer is the IoT Network Function Virtualization (IoT NVF) layer.

The Programmable Device layer can hold multiple logic clusters of devices sourced from multiple domains. We define a cluster of devices as being a group of devices that share communication capabilities and configurations. In an emergency, a cluster of devices from the same domain and sending the same alarm might have the communication switched off temporarily.

The IoT NFV layer enables infrastructure administrators to deploy on-demand services like M2M servers, IoT platforms, analytics engines on virtualized compute, networking, and storage nodes. An orchestrator can start new instances of these services and also inform services about existing, newly deployed instances of the same type or another inter-dependent type. The IoT NFV layer can be enriched with new services with defined interdependencies, i.e. topologies of NFV instances can be distributed over multiple datacentres by selecting the location at which a service instance should be placed.

The Flexible Logic Layer is comprised of communication controllers that can program devices such as smart sensors and gateways (programmable smart devices), and networking elements such as software switches in the case of Software Defined Networking. The Smart Device Communication Manager is the entity that programs the devices. It can advertise/force installation or updates of new communication software for the devices and configure their properties at runtime, without requiring a restart. The relationship with elements from Software Defined Networking (SDN) is envisioned here as status information about the forwarding nodes of the SDN and might be used as input to program the devices. The SDN can also use information about the forwarding nodes enforced on the smart devices to detect malfunctioning devices and report these to the Smart Device Communication Manager.

V. Architecture

A Proof-of-Concept (PoC) implementation was created to validate the suitability of a PaaS approach as applied to a multi-modal, geographically distributed sensor and service solution. In the PoC architecture (Fig. 3), a service related to a natural environment (Green City) and a service related to demand-side energy (Smart Energy) are implemented. These two services acquire data from energy and natural environment sensors. Actuation is also implemented for the energy service.

As back-end services, the OpenMTC M2M Server [15] and a Smart City Platform (SCP) enable standard compliant and real-world modelling of the IoT infrastructure. The OpenMTC is an implementation of the oneM2M [17] standard-compliant data management entity hosting collected data and actuating capabilities as smart objects (or resources). The SCP provides real-world modelling, linking the physical world infrastructure identities with the digital world identities (resources). The SCP also handles the geo-location of devices, collected data, and performs data statistics.

The role of the Smart Device Communication Manager is performed by the Open5GMTC [16] LWM2M Device Management, and monitors and manages the connectivity and communication between the smart devices. It includes a decision engine that receives triggers from the NFV Orchestrator or the deployed services directly in order to determine which service to allocate to a specific smart device.

An application for each supported domain is included on top of the SCP. The Smart City Application Green City for environmental monitoring is aimed at citizens who are interested in pollution and weather conditions. The User Mobile Application Smart Energy is destined for energy consuming users to check their consumption as reported by each smart device installed in a home and to switch appliances on or off to limit their consumption.

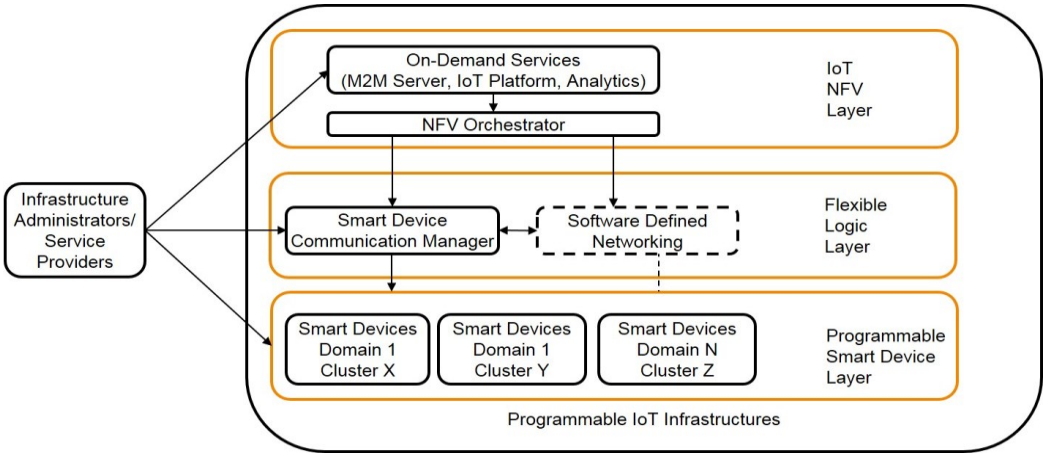

Fig. 2: End-to-End Programmable IoT Infrastructures

Three "Smart" Internet connected devices are used in the PoC smart energy domain with the ActivePlug and ActiveDIN being the energy sensing and controlling devices. The DIN provides three channels for sense and control (up to 20A each) while the Plug senses and controls only one channel (up to 16A). The DIN is installed in a typical electrical distribution board while the Plug is used with a South African wall-socket. The third device, ActiveGate, is a smart controller for managing a number of ActivePlug and ActiveDIN devices, by creating and maintaining a local 6LoWPAN mesh network (using 802.15.4 radios) with RPL (RFC6550) [10]. The ActiveGate enables data routing to and from higher-level control systems. In addition to the 6LoWPAN network, the ActiveGate also supports connectivity to the Internet through Wi-Fi, Ethernet and/or HSPA+3G.

Two device types support the environmental domain. The system is based on an opportunistic/delay-tolerant networking approach where sensing devices are installed in the city with the primary radio features in idle mode until a gateway device mounted on a public transportation vehicle gets close, wakes them up through a secondary low-consumption radio interface, and collects their information [8] - hence the name "wake-up sensors". No additional on-purpose infrastructure is needed. Sensing devices measure and report on environmental parameters such as temperature, humidity, light intensity, and ambient pressure. These devices are battery powered and rely on enhanced radio wake-up mechanisms for energy-efficiency. The DTN-based gateway device uses a radio wake-up transmitter to activate the sensing devices, a 802.15.4 radio interface to communicate, and a GPRS/3G modem to provide Internet connectivity and forward information to the remote platforms.

VI. Concept specific implementation extensions

The OpenSDNCore [18] ETSI NFV orchestrator was extended to have the services OpenMTC and SCP automatically deployed and their network interfaces reconfigured to new IP addresses. The FITeagle [19] FIRE tool was extended to support the above-mentioned backend services so that the PoC could be used by scientists.

The Open5GMTC Device Management Server is a hierarchical multi-processing LWM2M implementation based on libcoap [11]. It supports connectivity management of Wi-Fi and mobile profiles as well as bearer selection. The range of management objects was extended by adding an **own-defined Transport Management Policy** representing the service name, IP address and port of its instance, modelled as a smart object of the LWM2M protocol. Thus, the end devices can send notifications to Open5GMTC regarding the support for this object during registration and then receive updates when the server decides to enforce a new OpenMTC endpoint as M2M server for data collection.

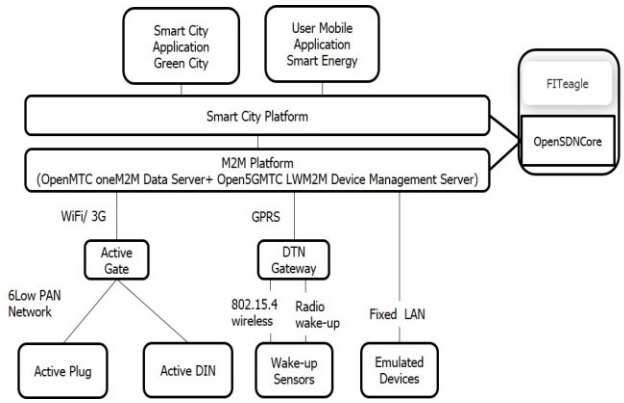

Fig. 3: Architecture

Both the DTN gateway and ActiveGate software were enhanced with LWM2M client software features to support registration, retrieval of device manufacturing information, connectivity information, and adaptation to new Transport Management Policies. The DTN gateway LWM2M client implementation uses the Python library txThings [9], while the ActiveGate uses Leshan LWM2M client [14].

The ActiveGate uses a variety of CoAP implementations: The JAVA-based Californium [12] library from Eclipse serves as basis for the interworking proxy between the devices and the M2M platform while the 6LoWPAN network uses the C-based Erbium [13] Contiki library. Finally, the C-based libcoap library is used in various diagnostic and management scripts.

VII. Experiments

Additional or replacement back-end services will be deployed in dynamic environments to deal with load, maintenance and other events. During these deployments, the edge devices and associated software stacks need to be updated with the new addresses of the cloud services.

The testbed is geographically distributed (Fig. 4) as follows: Environmental sensors are located in Spain (Barcelona), smart energy meters in South Africa (Pretoria), emulated devices in Germany (Berlin) and services are hosted in a cloud infrastructure in Berlin and Pretoria.

A. Static IoT infrastructure

The DTN gateway (I2CAT end device) physical device is an ODROID XU-3 machine running Ubuntu 14.04. The hardware is powered by a Samsung Exynos5422 with quad-core Cortex-A15 2.0Ghz and quad-core Cortex-A7 CPUs. It has 2GByte LPDDR3 RAM and is connected with a multi-slot class 12 GPRS interface to enable the communication to both the OpenMTC platform and the Device Manager. The cellular interface supports up to four timeslots in each, downlink and uplink, direction, of which up to five simultaneously. Depending on the coding scheme used, the available user throughput per slot is between 8kbps and 20kbps.

The maximum capacity of the link has been measured in both uplink and downlink directions using the iperf tool configured to send UDP packets. Results show a downlink throughput of 64.64kbps and 41.89 kbps for the uplink. The end-to-end delay results have been obtained from the ping tool configured to send 100 packets with a Maximum Transmission Unit (MTU) of 128 and 1500 bytes (TABLE I).

TABLE I. DTN GW - OPENMTC RTT

MTU	RTT (ms)			
	Average	*Min*	*Max*	*Standard Dev*
128	553.6	427.3	1096.8	85.1
1500	729.9	558.9	1296.6	116.7

The ActiveGate hardware includes an ODROID-U3 single board computer with an Exynos4412 Prime processor. This is a quad core ARM Cortex-A9 running at 1.7GHz with 2GB RAM and 8GB flash memory. Two custom designed boards interface with the ODROID to provide the 802.15.4 radio, LCD display, power, and general I/O capabilities. The operating system is Ubuntu 14.04.2 LTS in the Lubuntu (light-Ubuntu) variant. The Gate is connected to the rest of the system through a secure VPN and this mechanism is used to deliver updates.

Various performance measurements were taken on a typical PoC energy deployment comprising of one ActiveGate, one ActiveDIN, and two ActivePlugs. These included characteristics of the ActiveGate's processing capability as well as the 6LoWPAN's throughput to inform on the local scalability of the solution.

TABLE II. ACTIVEGATE PERFORMANCE

Resource Usage	Normal operation	Flood CoAP GETs
%CPU user code	1.47	7.686
%CPU kernel code	26.17	19.84
%CPU idle	72.43	72.618
%CPU IO waiting	0.09	0.036
%RAM M2M	1.4	1.4
%RAM Devices	0.03	0.03
6LoWPAN Tx Kb/s	0.544	35.2
6LoWPAN Rx Kb/s	0.456	34.7

TABLE II shows the resource usage of the ODROID-U3 during normal operation that is typically 1 CoAP transaction per second and flooded operation with an average transaction rate of 44.51 per second. The results show that the processing module has extensive resource margins and that the maximum throughput achieved on the 802.15.4 network (a total of 69.9Kb/s) is much lower than the theoretical limit of 250Kb/s for the 2.4GHz band.

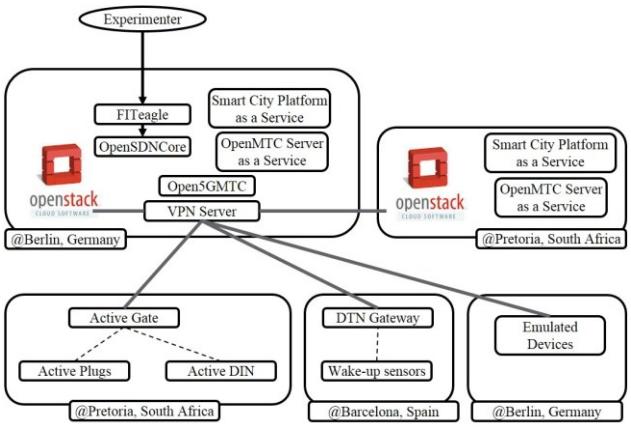

Fig. 4: PoC testbed

B. *Gradual scaling of IoT infrastructures*

The end devices register with the Open5GMTC Device Management (DM) Server. They automatically get a Transport Management Policy to send their data to the default M2M Middleware OpenMTC data server deployed for the PoC. The data is stored by the OpenMTC server. The default Smart City Platform (SCP) is notified about the new data. The SCP receives the data collected from the smart devices and the Green City App and the Smart Energy Mobile App then analyse and display these. The smart energy devices can be switched on or off using the Smart Energy Mobile App.

TABLE III presents the device management interface measurements that are related to the Open5GMTC server and located in the Berlin cloud. The interval considered spans from the moment the request is sent from the emitter until the reply is received, e.g. for registration the measurements were done on the client side and for read and write operations on the server side it includes the round-trip of messages and the processing on the other end.

TABLE III: DEVICE MANAGEMENT INTERFACE PERFORMANCE

Nb.	Average Duration (s)		
	Registration	*Read Device Info*	*Write Transport Management Policy*
DTN GW	0.5644	1.5109	1.809
Active Gate	0.334	0.4960	0.8423
Emulated Devices	0.1385	0.0059	0.0084

The read and write operations are sent in parallel to the endpoint. The DM server sends them asynchronously after the registration acknowledgement. The emulated devices are deployed on the Fraunhofer premises in Berlin, thus they reach the device management server with a minimum delay.

Localization of backend services on the same or different data-centers (potentially distributed over continents) can shed light on future federations and solutions for smart cities. Experiments of dynamic topologies using different service constellations (all in Berlin, all in Pretoria and hybrid) have been carried out to investigate the best outcome.

A topology of the OpenMTC data server and SCP is deployed via OpenSDNCore and FITeagle, the DM server is notified and updates the Transport Management Policy on the end devices. The devices update their target service for data delivery by means of IP addresses as communicated through the Transport Management Policy.

The results of the dynamic deployment are presented in TABLE IV. The PoC testbed in Fig. 4 depicts the geospatial combinations. Important to note is the location of the VPN server (hosted in Berlin). To ensure security, all communication is routed through this VPN server. This centralized server has proven to be a significant limiting factor. Deployment of instances in the Pretoria cloud indicates a

TABLE IV. DYNAMIC DEPLOYMENT AVERAGE DURATION (S)

Duration	*OpenMTC & SCP @Berlin*	*OpenMTC & SCP @Pretoria*	*OpenMTC @Berlin SCP @Pretoria*	*OpenMTC @Pretoria SCP @Berlin*
Services Deployment	440.8	883.5	568.3	903.7
OpenMTC-SCP bootstrap	5.4	3.9	183.8	209.7
ActiveGate OpenMTC Bootstrap	26.2	59.5	24.5	50.8
DTN Gw OpenMTC Bootstrap	0.73	1.2	1.1	1.7

significant delay compared to the deployment in Berlin. Previous tests have shown a small throughput of the VPN connection of approximately 5Mb/s between Berlin and Pretoria. This contrasts with throughput in excess of 700Mb/s in the Berlin cloud and approximately 90Mb/s between instances in the Pretoria cloud. Using a distributed VPN solution can improve the performance.

Device bootstrap (TABLE IV), for ActiveGate or DTN GW, measures the time after DM registration until all configurations are completed and the system is operational. In the case of environmental monitoring, a static seeding is performed in which all the DTN gateways and sensors are pre-created in the OpenMTC resource tree. The DTN GW OpenMTC bootstrap is a one-step request-reply procedure and it is clear that the duration is comparable for all deployments. The ActiveGate bootstrap indicates the consequence of having the distributed deployments and with traffic routed through a central VPN. Bootstrap time with OpenMTC in Pretoria is double that of OpenMTC hosted in Berlin (i.e. "close" to the VPN server). Similarly, bootstrap of the SCP from OpenMTC is severely affected when the SCP and OpenMTC are hosted in different clouds (Fig. 5). This is an important result and highlights the need for effective routing once scaling of devices and services are implemented, e.g. for a large Smart City.

VIII. CONCLUSIONS AND FUTURE WORK

Results of experimentation with PaaS on an end-to-end programmable IoT infrastructure have been presented. The intercontinental delay and the effect of security mechanisms such as VPNs are to be considered when tailoring topologies. Services should be placed as close as possible to the smart devices and roundtrips because of a centralized VPN solution. The delays in reconfiguring devices are valid for smart energy metering and environmental monitoring systems and the same concept could be applied to other domains. Latencies at the device level are typically not a factor beyond the local network.

Results from the TRESCIMO testbed indicate the viability of using a PaaS approach for scaling of services and devices for a large deployment typically associated with large Smart Cities. The testbed NFV Orchestrator allows for new services to be seamlessly added and instantiated, thus increasing the reproducibility of the concept, powered also by the trend of manufacturers adopting LWM2M device management protocol.

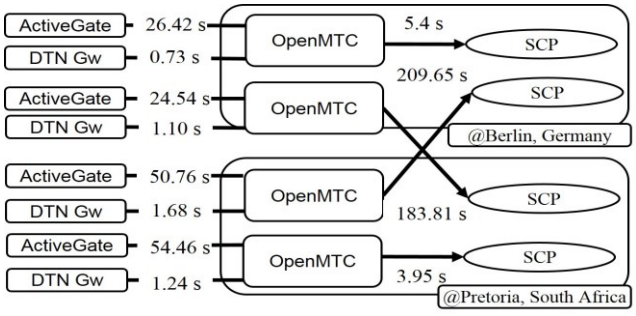

Fig. 5: Measurements overview

REFERENCES

[1] F. Li, M. Vogler, M. Claeßens and S. Dustdar, "Efficient and scalable IoT service delivery on Cloud", IEEE Sixth International Conference on Cloud Computing, 2013

[2] S. Ziegler, P. Kirsten, L. Ladid and A Skarmeta, "Evaluation and recommendations on IPv6 for the Internet of Things", ttp://iot6.eu/sites/default/files/IoT6%20-%20white%20paper%20-%20main%20conclusions.pdf

[3] Galis A., Clayman, S. ; Mamatas, L. ; Rubio Loyola, J. ; Manzalini, A. ; Kuklinski, S. ; Serrat, J. ; Zahariadis, T. "Softwarization of Future Networks and Services - Programmable Enabled Networks as Next Generation Software Defined Networks", IEEE SDN for Future Networks and Services (SDN4FNS), 2013.

[4] Alliance Open Mobile, "Lightweight Machine to Machine Architecture V1.0." 2012.

[5] Z. Shelby, K. Hartke, and C. Bormann, "RFC 7252: The Constrained Application Protocol (CoAP)." 2014.

[6] Z. Shelby, " RFC 6690: Constrained RESTful Environments (CoRE) Link Format.", August 2012.

[7] Testbeds for Reliable Smart City Machine-to-Machine Communication (TRESCIMO). [Online] Available: www.trescimo.eu

[8] Elmangoush, A. Steinke, R. ; Catalan, M. ; Corici,A. ; Magedanz, T. ; Oller, J."Interconnecting Standard M2M Platforms to Delay Tolerant Networks", FiCloud - International Conference on Future Internet of Things and Cloud, 2014

[9] txThings- CoAP library for Twisted framework. https://github.com/siskin/txThings

[10] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander, "RFC6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks." 2012

[11] libcoap: C-Implementation of CoAP. http://sourceforge.net/projects/libcoap/

[12] Californium (Cf) core. https://eclipse.org/californium/

[13] Erbium (Er) REST Engine and CoAP Implementation for Contiki. https://people.inf.ethz.ch/mkovatsc/erbium.php

[14] Leshan, OMA Lightweight M2M server and client Java implementation, https://github.com/eclipse/leshan

[15] OpenMTC, ETSI M2M & oneM2M implementation, www.openmtc.org

[16] Open5GMTC, Prototyping future M2M, www.open5gmtc.org

[17] oneM2M, standardization forum, onem2m.org

[18] OpenSDNCore, SDN & ETSI NFV prototype, www.opensdncore.org

[19] FITeagle, www.fiteagle.org

[20] ETSI NFV MANO, http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[21] FIRE GENI, http://groups.geni.net/geni/wiki/FIREGENI

[22] Smart Santander, http://www.smartsantander.eu